Difference between ForEach and ForEach-Object in powershell

Asked 4 years, 4 months ago Active 6 months ago Viewed 31k times

24

5

Is there any difference between `ForEach` and `ForEach-Object` ?

I have a small code like this, works fine

```
$txt = Get-Content 'C:\temp\000.txt'
$result = foreach ($line in $txt) {$line.replace(".ini","")}
$result | out-file 'c:\temp\001.txt'
```

But if i use 'ForEach-Object', I got errors....

```
$txt = Get-Content 'C:\temp\000.txt'
$result = foreach-object ($line in $txt) {$line.replace(".ini","")}
$result | out-file 'c:\temp\001.txt'
```

Why ? and how to output the loop results by using `ForEach-Object`

powershell

asked Mar 19 '15 at 15:18
Root Loop
1,459 8 30 46

1 social.technet.microsoft.com/Forums/en-US/... – dugas Mar 19 '15 at 15:46

4 Answers

19

✓

`foreach` is an alias of `ForEach-Object` but it appears to also be a keyword (which is confusing).

The `foreach ($<item> in $<collection\>){<statement list>}` syntax you are using is `help about_foreach` .

The `foreach` as `ForEach-Object` alias is `help ForEach-Object` .

The keyword `foreach` operates over each `$<item>` in the `$<collection>` as given in the `()` bit.

edited Mar 19 '15 at 15:46 answered Mar 19 '15 at 15:25
Etan Reisner
60.8k 5 55 93

well, if i change the code to `ForEach-Object` , I get error `Unexpected token 'in' in expression or statement. Missing closing ')' in expression. Unexpected token ')' in expression or statement.` .... – Root Loop Mar 19 '15 at 15:40

1 Right. Because `ForEach-Object` doesn't use that syntax. Look at the help documents I listed. – Etan Reisner Mar 19 '15 at 15:45

16

They're different commands for different purposes. The ForEach-Object **cmdlet** is used in the pipeline, and you use either $PSItem or $_ to refer to the current object in order to run a {scriptblock} like so:

```
1..5 | ForEach-Object {$_}

>1
>2
>3
>4
>5
```

Now, you can also use a very similiar looking **keyword**, ForEach, at the beginning of a line. In this case, you can run a {scriptblock} in which you define the variable name, like this:

```
ForEach ($number in 1..5){$number}
>1
>2
>3
>4
>5
```

The core difference here is where you use the command, one is used in the midst of a pipeline, while the other starts its own pipeline. In production style scripts, I'd recommend using the ForEach keyword instead of the cmdlet.

edited Mar 19 '15 at 16:08 answered Mar 19 '15 at 15:54
FoxDeploy
6,406 2 17 31

Keywords : ForEach-Object used in the pipeline. nice. – Root Loop Mar 19 '15 at 15:58

7

Both the previous answers are correct, but https://blogs.technet.microsoft.com/heyscriptingguy/2014/07/08/getting-to-know-foreach-and-foreach-object/ has both a good summary:

> When you are piping input into ForEach, it is the alias for ForEach-Object. But when you place ForEach at the beginning of the line, it is a Windows PowerShell statement.

and more details:

> The ForEach statement loads all of the items up front into a collection before processing them one at a time. ForEach-Object expects the items to be streamed via the pipeline, thus lowering the memory requirements, but at the same time, taking a performance hit.

He then includes some performance measurements and concludes:

> So which one do you use? Well, the answer is, "It depends." You can iterate through a collection of items by using either the ForEach statement or the ForEach-Object cmdlet. ForEach is perfect if you have plenty of memory, want the best performance, and do not care about passing the output to another command via the pipeline. ForEach-Object (with its aliases % and ForEach) take input from the pipeline. Although it is slower to process everything, it gives you the benefit of Begin, Process, and End blocks. In addition, it allows you to stream the objects to another command via the pipeline. In the end, use the approach that best fits your requirement and the capability of your system.

answered Sep 7 '18 at 23:53
Orangutech
338 3 9

0

Apart from the technical differences that have been mentioned before, here are some practical differences, for sake of completeness (see also):

1.) In a pipeline, you do not know the total count of the processed items. This is a situation where you would opt to acquire the complete list first and then do a foreach-loop.

Example:

```
$files = gci "c:\fakepath"
$i = 0
foreach ($file in $files) {
    $i++
    Write-Host "$i / $($files.Count) processed"
}
```

2.) With an existing list, the `foreach` loop is faster than then pipeline version, because the script

```
$items = 0..100000
Measure-Command { $items | ForEach-Object { $_ } }
# ~500ms on my machine
Measure-Command { foreach ($i in $items) { $i } }
# ~70ms on my machine
```

answered Jan 24 at 11:20

marsze
6,096 3 21 43